Low-memory Lagrangian Relaxation Methods for Sensor Placment in Municipal Water Networks

Jonathan W. Berry* Erik Boman* Cynthia A. Phillips*
Lee Ann Riesen*

Abstract

Placing sensors in municipal water networks to protect against a set of contamination events is a classic p-median problem for most objectives when we assume that sensors are perfect. Many researchers have proposed exact and approximate solution methods for this p-median formulation. For full-scale networks with large contamination event suites, one must generally rely on heuristic methods to generate solutions. These heuristics provide feasible solutions, but give no quality guarantee relative to the optimal placement.

In this paper we apply a Lagrangian relaxation method in order to compute lower bounds on the expected impact of suites of contamination events. In all of our experiments with single objectives, these lower bounds establish that the GRASP local search method generates solutions that are provably optimal to to within a fraction of a percentage point. Our Lagrangian heuristic also provides good solutions itself and requires only a fraction of the memory of GRASP.

We conclude by describing two variations of the Lagrangian heuristic: an aggregated version that trades off solution quality for further memory savings, and a multi-objective version which balances objectives with additional goals.

1 Introduction

A key component of contaminant warning system design for municipal water distribution networks is the intelligent location of sensors. A typical process for determining these locations begins with the computation of an extensive suite of simulated contamination incidents using, for example, EPANET [21]. The results of these simulations feed optimization algorithms that attempt to minimize the expected impact of an incident in the input suite. In its most familiar form, this problem reduces to a "p-median" problem (described in Section 2). This theoretically hard problem typically defies solution methods that attempt to find provably optimal solutions as the size of the network grows past tens of thousands of nodes.

*Sandia National Laboratories, Mail stop 1318, P.O. Box 5800, Albuquerque, NM, 87185-1318, USA. {jberry, egboman, caphill, lafisk}@sandia.gov. Sandia is a multipurpose laboratory operated by Sandia Corporation, a Lockheed-Martin Company, for the United States Department of Energy's National Nuclear Security Admininstration under contract DE-AC04-94AL85000.

However, experimental results shows that local search heuristics such as GRASP [20] seem to readily and accurately solve those p-median instances that arise in the context of water sensor placement. In previously published work [9], we showed for some moderate-sized instances that the solutions produced by GRASP were in fact provably optimal. However, we were required to use an exact solution method called Mixed-Integer Programming (MIP) to achieve those results. As water distribution network sizes increase indefinitely, though, MIP can no longer be expected to find optimal solutions. There are two reasons for this. The first is that solving very large MIP models to optimality is a well-known hard problem. With that said, the MIP model can be relaxed into a linear programming (LP) model that provides lower bounds on solutions quality and is computationally tractable. Therein lies the second, and more important reason: even solving the LP requires large amounts of computer memory.

In this paper we apply a classical method called Lagrangian relaxation to the problem of lower bound computation. The method we apply exploits mathematical structure in the p-median problem in order to reduce the memory requirements. In Lagrangian relaxation, we remove some of the constraints, instead applying pressure to satisfy them by adding penalties to the objective function when we fail to meet them. We consider Barahona and Chudak's Lagrangian method for p-median problems [3]. This uses Barahona and Anbil's Volume algorithm [2], a subgradient method that produces a series of valid lower bounds, converging to the optimal linear-programming solution. Furthermore, we can use a specialized form of randomized rounding to compute a feasible sensor placement that is biased by this fractional LP solution. Thus the Lagrangian code is a true solver rather than just a lower bound engine.

The Lagrangian relaxation method requires space proportional to the input data, while other previous methods, including LP, require space proportional to the square of the input data in the worst case. For large problems, the Lagrangian relaxation method requires an order of magnitude less space than LP methods and even an order of magnitude less space than fast implementations of local-search methods such as GRASP. The lower bounds are extremely close to the LP bounds and the solutions are usually within 10% of the best known solution.

2 Background

Researchers have developed a variety of technical approaches for solving sensor placement problems including mixed-integer programming (MIP) models [5, 4, 16, 15, 19, 22], combinatorial heuristics [13, 14, 18], and general-purpose metaheuristics (e.g., [18]).

A common feature of most sensor placement formulations is that they rely either directly or indirectly on contaminant transport simulation models. Simulation tools, like EPANET [21], perform extended-period simulation of the hydraulic and water quality behavior within pressurized pipe networks. These models can be used to evaluate the expected flow in water distribution systems, and they can model the transport of contaminants and related chemical interactions. Thus, a water utility can assess risks to their distribution network by considering simulations of an ensemble of contamination incidents, which reflect the impact of contamination at different locations, times of the day, etc.

A key limitation of early sensor placement formulations is that they incorporate contamination transport simulation results indirectly. Consequently, the optimized value of the final solution may not accurately approximate a risk assessment performed with contaminant transport simulations. We have proposed a mixed-integer programming (MIP) model that resolves this difficulty by directly integrating contaminant transport simulation results [8, 5]. The MIP objective exactly captures water utilities' current risk metrics. Furthermore, this model can minimize a variety of different design objectives simply by integrating different statistics from the simulation results. This model assumes that a potentially large number of contamination incidents can be simulated, but these simulations are preprocessing steps that can be done in advance of the optimization process. Thus, the time needed for simulation does not impact the time spent performing sensor placement.

Our MIP formulation for sensor placement is:

$$
\text{(SP)} \quad \text{minimize} \quad \sum_{a \in \mathcal{A}} \alpha_a \sum_{i \in \mathcal{L}_a} d_{ai} x_{ai}
$$
$$
\text{where} \quad \begin{cases} \sum_{i \in \mathcal{L}_a} x_{ai} = 1 & \forall a \in \mathcal{A} \\ x_{ai} \leq s_i & \forall a \in \mathcal{A}, i \in \mathcal{L}_a \\ \sum_{i \in L} s_i \leq p & \\ s_i \in \{0,1\} & \forall i \in L \\ 0 \leq x_{ai} \leq 1 & \forall a \in \mathcal{A}, i \in \mathcal{L}_a \end{cases}
$$

This MIP minimizes the expected impact of a set of contamination incidents defined by $\mathcal{A}$. For each incident $a \in \mathcal{A}$, α_a is the weight of incident a, frequently a probability. The EPANET simulator reports contamination levels at a set of *locations*, denoted L, where a location refers to network junction. For each incident a, $\mathcal{L}_a \subseteq L$ is the set of locations that can be contaminated by a. Thus a sensor at a location $i \in \mathcal{L}_a$ can detect contamination from incident a at the time contamination first arrives at location i. Each incident is *witnessed* by the first sensor to see it. For each incident $a \in \mathcal{A}$ and location $i \in \mathcal{L}_a$, d_{ai} defines the impact of the contamination incident a if it is witnessed by location i. This impact measure assumes that as soon as a sensor witnesses contamination, then any further contamination impacts are mitigated (perhaps after a suitable delay that accounts for the response time of the water utility). The s_i variables indicate where sensors are placed in the network, subject to a budget p, and the x_{ia} variables indicate whether incident a is witnessed by a sensor at location i.

We may not be able to witness all contamination incidents with a given set of sensors. To account for this, L contains a *dummy* location. This dummy location is in all subsets $\mathcal{L}_a$. The impact for this location is the impact of the contamination incident after the entire contaminant transport simulation has finished, which corresponds to the impact that would occur without an online CWS.

Remarkably, SP is identical to the well-known p-median facility location problem [17]. In the p-median problem, p facilities (e.g., central warehouses) are to be located on m potential sites such that the sum of distances d_{ai} between each of n customers (e.g., retail outlets) and the nearest facility i is minimized. In comparing SP and p-median problems, we observe equivalence between (1) sensors and facilities, (2) contamination incidents and customers, and (3) contamination impacts and distances. While SP allows placement of *at most* p sensors, p-median formulations generally enforce placement of all p facilities; in practice, the distinction is irrelevant unless p approaches the number of possible locations.

3 Lagrangian Heuristic

We now describe a Lagrangian relaxation model for the sensor placement problem. As with all Lagrangian relaxation, we remove some of the constraints, leaving behind a problem that is easy to solve. We apply pressure to satisfy the constraints we have relaxed by adding penalties to the objective function. These penalties are proportional to the constraint violations. Thus there is no penalty if a constraint is met, a small penalty for a small violation, and a larger penalty for a larger violation.

We relax the first set of constraints in the SP formulation, those that require each incident is witnessed by some sensor; recall that this might be the *dummy* sensor that indicates a failure to detect the incident. This constraint is written as an equality, because that is a more efficient integer programming formulation. However, the difficult part of the constraint is insuring that at least one sensor witnesses each incident. The objective will prevent over-witnessing, so for the sake of the Lagrangian relaxation, we consider these constraints to be inequalities. For some incident a, this constraint is violated for a proposed setting of the s_i and x_{ai} variables if $\sum_{i \in \mathcal{L}_a} x_{ai} < 1$, giving a violation of $1 - \sum_{i \in \mathcal{L}_a} x_{ai}$. We weight each such violation with its own Lagrangian multiplier λ_a, which allows us to penalize some violations more than others. Adding a penalty term $\lambda_a - \lambda_a \sum_{i \in \mathcal{L}_a} x_{ai}$ to the objective for each incident a, the Lagrangian model becomes:

$$\text{(LAG)} \quad \text{minimize} \quad \sum_{a \in \mathcal{A}} \left(\alpha_a \sum_{i \in \hat{\mathcal{L}}_a} (d_{ai} - \lambda_a) x_{ai} \right) + \sum_{a \in \mathcal{A}} \alpha_a \lambda_a$$

$$\text{where} \quad \begin{cases} x_{ai} \le s_i & \forall a \in \mathcal{A}, i \in \hat{\mathcal{L}}_a \\ \sum_{i \in L} s_i \le p & \\ s_i \in \{0, 1\} & \forall i \in L \\ 0 \le x_{ai} \le 1 & \forall a \in \mathcal{A}, i \in \hat{\mathcal{L}}_a \end{cases}$$

For a fixed set of λ_a, we can compute the optimal value of LAG in linear space and near-linear time using a slight variation on the method described by Avella, Sassano, and Vasil'ev [1]. The optimal solution to LAG gives a valid lower bound on the value of an optimal solution to the p-median (SP) problem. This is because any feasible solution to the p-median problem is feasible for LAG. It has a zero violation for each of the lifted constraints and a value equal to the original p-median value.

4 Solution Methods

We consider two major solution strategies for the sensor-placement problem: Lagrangian and local search.

The local search method was described in [9]. Berry et. al. applied a variation of Resende and Werneck's GRASP heuristic for the p-median problem [20] to the sensor-placement problem. This heuristic operates by iteratively improving an initial random sensor placement, and then evaluating every "neighboring" solution. The best neighboring solution then becomes the current solution, and we repeat the search process until the current solution is locally optimal. Our neighborhood for a sensor placement S with p sensors is any other sensor placement that agrees with S on $p-1$ placements. So we find neighbors by moving a single sensor to another location.

We consider two variants of the local search method: dense and sparse. This refers only to the representation of the impact matrices. The sparse version uses less space at the expense of time. The internal local search method uses specialized GRASP code, which in its fastest form uses dense data structures (like two-dimensional matrices proportional to n^2 where n is the number of nodes). This is common to both the sparse and dense implementations we used.

The other major strategy is to solve the Lagrangian model given in Section 3. We use the Lagrangian-based lower-bounding method for the p-median problem described by Avella, Sassano, and Vasil'ev [1]. They give a Lagrangian model for which one can compute the optimal solution, given a set of Lagrangian multipliers, in linear space and near-linear time. More specifically, the Lagrangian-based bounding procedure requires $O(n+D)$ space, where n is the number of sensor locations and D is the total number of impacts. This is an asymptotically optimal memory requirement for an in-core implementation. Barahona and Chudak [3] give a Lagrangian formulation for the related unconstrained facility location problem, where one balances a facility opening cost with the service costs rather than limiting the number of facilities. Barahona and Chudak detail how to use subgradient search, specifically Barahona and Anbil's Volume algorithm [2], to find Lagrangian multipliers that produce progressively higher lower bounds. This method begins with $\lambda_a = 1$ for all incidents a, solves the relaxed problem, then iteratively updates the multipliers, increasing the multipliers in proportion to the violation. The updates require space and time linear in the number of variables. We modified the Vol unconstrained facility location code, available in the COIN-OR repository [10] for the p-median problem. This will converge to an optimal solution for the p-median problem. This search converges to a set of Lagrangian multipliers for which the optimal solution to our relaxed problem is an optimal solution to the p-median linear-programming (LP) relaxation.

Given a fractional solution to the p-median LP, we can treat the fractional values as probabilities and select sensors randomly according to this probability. However, one is unlikely to get precisely p sensors this way. We use the method of Berry and Phillips [6] for efficiently sampling over the "lucky" distribution where we select precisely k sensors. If necessary, we then select the dummy location.

We consider two ways to present a p-median problem to the Lagrangian solver. In the first, we present the set of locations and possible witnesses as described. This is the *non-aggregated* input method. In the second method, we *aggregate* witnesses with identical impacts into *superlocations*. The superlocation can cover an incident with one node rather than the whole set. As Berry et al described [9, 7], an integer program benefits from this space reduction while still selecting at least one real location for each selected superlocation. The Lagrangian method is not aware of superlocation structure, so we must select a real location after the computation. We currently simply select the first one. In Section 8 we discuss possible ways to do this better.

Method	net 1 1 ev/day	net 2 1 ev/day	net 3 1 ev/day	net 1 4 ev/day	net 2 4 ev/day	net 3 4 ev/day
Lag Agg	0.27	18	83	1.1	70	200
Lag No Agg	0.32	62	280	2.4	230	1000
Sparse LS	0.23	25	220	1.1	95	720
Dense LS	0.097	7.9	86	0.34	27	290

Table 1: Table showing average runtime in seconds for the various solvers. Lag Agg and Lag No Agg means Lagrangian with and without aggregation respectively. LS means local neighborhood search. ev/day means number of events per day per non-zero demand node.

5 Experimental Methodology

Our experiments compare the performance of the Lagrangian solver with and without aggregation to the local-search heuristic with either a dense or sparse impact-value representation. We considered three water networks for our experiments: a "small" instance (network 1) with 410 nodes, a "medium" instance (network 2) with 3358 nodes, and a "large" instance (network 3) with 11575 nodes. These are the same three instances used in Berry et al. [9]. As in that work, for each instance we created injection ensembles containing injections at each non-zero demand node (105, 1621, and 9705 nodes respectively) for either a single start time or for four equally-spaced start times. We consider biological contamination incidents for which the injection duration is 24 hours, and the inject rate is a large number of cells per liter (which we omit intentionally).

We considered three impact metrics/objectives: volume of contaminant removed from the network via demand, mass of contaminant similarly removed from the network, and extent of pipe contamination. For the four-event-per-day runs, we considered only mass consumed and extent of contamination. Thus we had 15 test cases (3 networks with 5 combinations of incidents and objective). For all our experiments we assume no detection delay and we placed 10 sensors.

6 Results

Tables 1, 2, and 3 show the performance of the solution methods with respect to runtime, memory, and closeness to optimality respectively. The local-search heuristic always found the optimal solution, as proven by also solving the problems with the CPLEX 9.0 or 10.0 integer programming solver (for all but one of the largest problems). The non-aggregated Lagrangian method (with rounding) was usually within 2% of optimal. The aggregated Lagrangian method, which had no access to the underlying actual locations, did signficantly worse in minimizing impact. However, its objective value was usually within a factor of 2 of the optimal.

The smallest network (network 1), was so small that constant-factor terms dominated the space requirements for the methods. For this network, the local-search heuristics required less space than the Lagrangian methods. However, by network 2, the assmptotic effects are starting to domingate, so Lagrangian methods required less space than the local-search methods and by network 3, especially with four incidents per non-zero-demand node per day, the Lagrangian methods required much less space. In these largest problems, even the non-aggregated Lagrangian method required almost an order of magnitude less space than the dense local search.

The non-aggregated Lagrangian method ran about three-to-five times slower than the dense local search. The aggregated version ran about as fast as dense local search.

Method	net 1 1 ev/day	net 2 1 ev/day	net 3 1 ev/day	net 1 4 ev/day	net 2 4 ev/day	net 3 4 ev/day
Lag Agg	12.1	29.4	98.0	13.6	77.3	207
Lag No Agg	12.5	81.4	330	15.6	290	1150
Sparse LS	4.29	165	1530	7.37	387	2900
Dense LS	3.75	153	2150	8.38	604	8170

Table 2: Table showing average memory in MB for the various solvers. Lag Agg and Lag No Agg means Lagrangian with and without aggregation respectively. LS means local neighborhood search. ev/day means number of events per day per non-zero demand node.

Overall, the non-aggregated Lagrangian solver is a good choice for problems up to roughtly three times too large to fit on a standard workstation. The aggregated version can give some solution to problems up to an order of magnitude too large.

Method	net 1 1 ev/day	net 2 1 ev/day	net 3 1 ev/day	net 1 4 ev/day	net 2 4 ev/day	net 3 4 ev/day
Lag Agg	39	85	160	37	48	59
Lag No Agg	2.0	13	0.71	0	1	2.4

Table 3: Table showing average percent over optimal for the Lagrangian solvers. Lag Agg and Lag No Agg means Lagrangian with and without aggregation respectively. LS means local neighborhood search. ev/day means number of events per day per non-zero demand node. The local-search solvers gave the optimal solution on all problems.

Lower Bound Results For the 15 problems we considered, in the worst case, the lower bound was 96.11 percent of the optimal. In all the other cases, the lower bound was at least 99.99 percent of the optimal value. Thus the Lagrangian method essentially proved the optimality of the local search heuristic for these cases. Thus the Lagrangian methods can provide strong lower bounds in an acceptable amount of time using vastly less space than linear-programming solvers.

7 Multiple Objectives

In this section we consider the sensor placement problem with multiple objectives. One might want to optimize both population impacts and network impacts rather than just one and generally one may want to optimize many objectives simultaneously. Any solution that is not *dominated* could be interesting. A solution is dominated if there is another solution that is at least as good in every objective, and strictly better in at least one metric.

The Lagrangian method described in Section 3 handles multiple objectives easily and naturally. Suppose we have a goal value g for a secondary objective. That is, we wish to enforce $\sum_{a,i\in\mathcal{L}_a} d'aix_{ai} \leq g$, for an objective with impacts d'. The violation for this constraint is $\sum_{a,i\in\mathcal{L}_a} d'aix_{ai} - g$. We can weight this violation with its own Lagrangian multiplier (weight) and add it to the objective as well. We can then solve using the same methods. The solution may violate the goal, but the multipliers puts pressure on the solver to meet the goal. Our current implementation does not allow aggregation when there are side constraints. There is no obvious way to aggregate in a way that works well for all impacts.

The local-search solver in the SPOT toolkit [12] also supports such side constraints. However, this solver treats the constraints as hard. It will not move to a neighboring solution if the side constraint is violated. Otherwise, the search proceeds as before. The solver does not support more than one side constraint, in part because even three side constraints can lead to frequent failures to find a feasible solution.

Using the same set experimental data as described in Section 5, we created multi-objective variants as follows. We solved each of the single-objective problems to optimality using an integer programming solver and evaluated that sensor placement for all three metrics. For each problem (network, incident set), let b_m be the worst value of any of these sensor placements for metric m and let β_m be the optimal for metric m. Let range $r = b_m - \beta_m$. For that problem and metric, we considered four goals: the optimal β_m, and the optimal plus $.25r$, $.5r$, and r. In the last case, we have a reasonably relaxed goal equivalent to what a solver found when ignoring metric m. We considered problems with single side constraints and, for the Lagrangian solver, with two side constraints.

Space constraints prohibit an extensive discussion of this experiment, but we now summarize the main points. For networks 1 and 2, the Lagrangian methods found fewer non-dominated solutions than the local search. However, those it found were generally a little more balanced across the objectives and had a little better average impact across all metrics. For network 3, the local search methods generally did better when they solved, but in 2 of the 24 problems the local search methods failed to find a solution even though one existed. Furthermore the local search methods do not use the GRASP optimizations with side consraints. Therefore by network 3, these searches took much longer than the Lagrangian methods. For network 3, 1 event-per-node-per-day problems, local search for the dense method required 11 hours on average where (non-aggregated) Lagrangian required 15 minutes even when using two goals. The sparse heuristic was effectively too slow to use. And even with 2 side constraints, which requires 3 sets of impact data, Lagrangian required about 700MB of memory while the dense heuristic with one side constraint required 3.2GB

8 Further Work

There are a number of ways we might improve the performance of aggregated Lagrangian. For example, instead of arbitrarily selecting a real location from each superlocation, we can find an optimal hitting set (e.g. using [11]). This finds a smallest number of locations to cover all the superlocations. Though hitting set is an NP-complete problem, and is even theoretically hard to approximate, it will likely be tractible for this size problem, since we usually do not place too many sensors. If we need fewer than p sensors to cover the superlocations, we can add the remainder greedily to maximally reduce impact. This will likely be important as the number of sensors increases.

We might improve the performance of the multiobjective Lagrangian method by handling the tiny number of multipliers for the goal constraints differently from the way we handle the multipliers for the witness constraints.

References

[1] P. Avella, A. Sassano, and I. Vasil'ev. Computational study of large-scale p-median problems. *Mathematical Programming*, 109(1):89–114, January 2007.

[2] F. Barahona and R. Anbil. The volume algorithm: producing primal solutions with a subgradient method. *Mathematical Programming*, 87(3):385–399, May 2000.

[3] F. Barahona and F. Chudak. Near-optimal solutions to large-scale facility location problems. *Discrete Optimization*, 2:35–50, 2005.

[4] J. Berry, L. Fleischer, W. E. Hart, C. A. Phillips, and J.-P. Watson. Sensor placement in municipal water networks. *J. Water Planning and Resources Management*, 131(3):237–243, May 2005.

[5] J. Berry, W. E. Hart, C. A. Phillips, and J. Uber. A general integer-programming-based framework for sensor placement in municipal water networks. In *Proc. World Water and Environment Resources Conference*, 2004.

[6] J. Berry and C. Phillips. Randomized rounding for sensor placement problems. In preparation. To be submitted., 2007.

[7] Jonathan Berry, Robert D. Carr, William E. Hart, and Cindy A. Phillips. Scalable water sensor placement via aggregation. In *Proc. Water Distribution System Symposium*, 2007.

[8] Jonathan Berry, William E. Hart, Cynthia E. Phillips, James G . Uber, and Jean-Paul Watson. Sensor placement in municiple water networks with temporal integer prog ramming models. *J. Water Resources Planning and Management*, 132(4):218–224.

[9] Jonathan Berry, William E. Hart, Cynthia E. Phillips, James G . Uber, and Jean-Paul Watson. Sensor placement in municiple water networks with temporal integer prog ramming models. *J. Water Resources Planning and Management*, 132(4):218–224, 2006.

[10] Computational INfrastructure for Operations Research home page. `http://www.coin-or.org/`.

[11] Henning Fernau. Parametrized algorithms for hitting set: the weighted case. In *Proc. Conference on Algorithms and Complexity, Lecture Notes on Computer Science 3998*, 2006.

[12] William E. Hart, Jonathan Berry, Regan Murray, Cindy A. Phillips, Lee Ann Riesen, and Jean-Paul Watson. SPOT: A sensor placement optimization toolkit for drinking water contaminant warning system design. Technical Report SAND2007-4393 C, Sandia National Laboratories, 2007.

[13] A. Kessler, A. Ostfeld, and G. Sinai. Detecting accidental contaminations in municipal water networks. *Journal of Water Resources Planning and Management*, 124(4):192–198, 1998.

[14] A. Kumar, M. L. Kansal, and G. Arora. Discussion of 'detecting accidental contaminations in municipal water networks'. *Journal of Water Resources Planning and Management*, 125(4):308–310, 1999.

[15] B. H. Lee and R. A. Deininger. Optimal locations of monitoring stations in water distribution system. *Journal of Environmental Engineering*, 118(1):4–16, 1992.

[16] B. H. Lee, R. A. Deininger, and R. M. Clark. Locating monitoring stations in water distribution systems. *Journal, Am. Water Works Assoc.*, pages 60–66, 1991.

[17] P.S. Mirchandani and R.L. Francis, editors. *Discrete Location Theory*. John Wiley and Sons, 1990.

[18] A. Ostfeld and E. Salomons. Optimal layout of early warning detection stations for water distribution systems security. *Journal of Water Resources Planning and Management*, 130(5):377–385, 2004.

[19] M. Propato, O. Piller, and J. Uber. A sensor location model to detect contaminations in water distribution networks. In *Proc. World Water and Environmental Resources Congress*. American Society of Civil Engineers, 2005.

[20] M.G.C. Resende and R.F. Werneck. A hybrid heuristic for the p-median problem. *Journal of Heuristics*, 10(1):59–88, 2004.

[21] L. A. Rossman. The EPANET programmer's toolkit for analysis of water distribution systems. In *Proceedings of the Annual Water Resources Planning and Management Conference*, 1999. Available at `http://www.epanet.gov/ORD/NRMRL/wswrd/epanet.html`.

[22] J.-P. Watson, H. J. Greenberg, and W. E. Hart. A multiple-objective analysis of sensor placement optimization in water networks. In *Proc. World Water and Environment Resources Conference*, 2004.